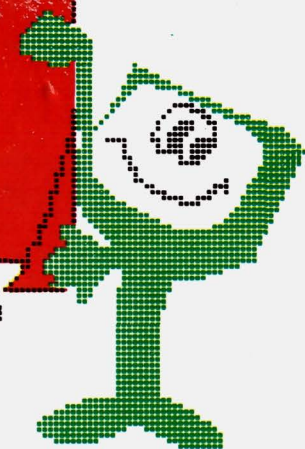
20 VIDEOLEZIONI DI BASIC
PER IMPARARE COL VIC 20

GRUPPO
EDITORIALE
JACKSON

4

COMMODORE
VIC 20

VIDEO BASIC VIC 20
Pubblicazione quattordicinale
edita dal Gruppo Editoriale Jackson
Direttore Responsabile:
Giampietro Zanga
Direttore e Coordinatore
Editoriale: Roberto Pancaldi
Autore: Softidea via Indipendenza 88 Como
Redazione software:
Francesco Franceschini, Enrico Braglia,
Fabio Calanca
Segretaria di Redazione:
Marta Menegardo
Progetto grafico:
Studio Nuovaidea - Via Longhi 16 - Milano
Impaginazione:
Silvana Corbelli
Illustrazioni:
Cinzia Ferrari, Silvano Scolari
Fotografie:
Marcello Longhini
Distribuzione: SODIP
Via Zuretti, 12 - Milano
Fotocomposizione: Lineacomp S.r.l.
Via Rosellini, 12 - Milano
Stampa: Grafika '78
Via Trieste, 20 - Pioltello (MI)
Direzione e Redazione:
Via Rosellini, 12 - 20124 Milano
Tel. 02/6880951/5

Autorizzazione alla pubblicazione Tribunale di Milano n° 422 del 22-9-1984
Spedizione in abbonamento postale Gruppo II/70 (autorizzazione della Direzione Provinciale delle PPTT di Milano).
Prezzo del fascicolo L. 8.000
Abbonamento comprensivo di 5 raccoglitori L. 165.000
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson S.r.l. - Via Rosellini, 12 20124 Milano, mediante emissione di assegno bancario o cartolina vaglia oppure utilizzando il c.c.p. n° 11666203.
I numeri arretrati possono essere richiesti direttamente all'editore inviando L. 10.000 cdu. mediante assegno bancario o vaglia postale o francobolli.
Non vengono effettuate spedizioni contrassegno.

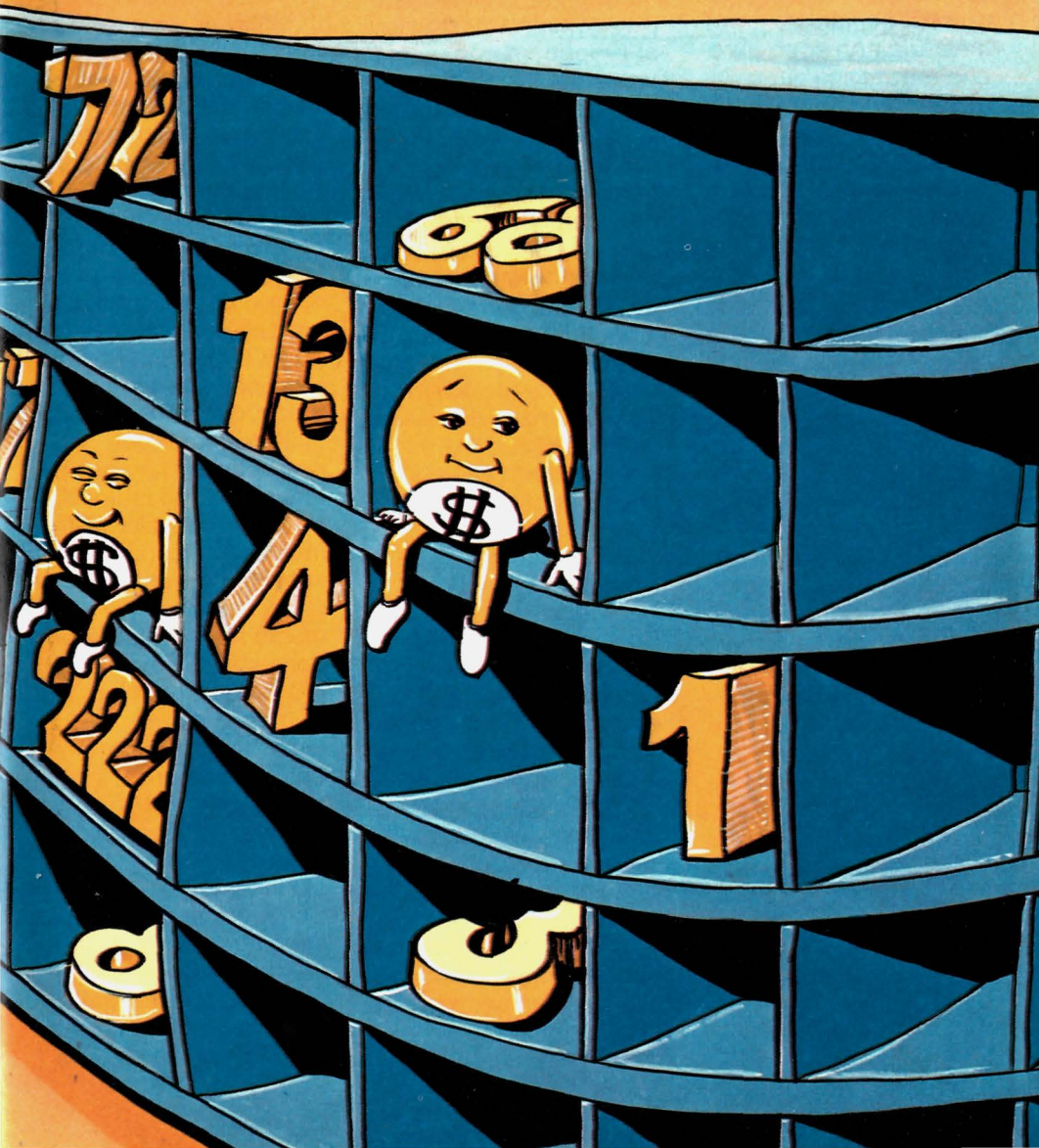
Gruppo Editoriale Jackson

SOMMARIO

Introduzione

Saper ricordare è fondamentale per noi come per lui (il tuo computer): senza "memoria" è una scatola, vuota, inutile. Chi gli "ricorderebbe" cosa fare quando lo accendi o gli dai le istruzioni BASIC.? E dove potrebbe prendere appunti (memorizzare) su quello che sta facendo? Solo la memoria, una memoria prodigiosa che non dimentica mai (ROM) o che è disponibile e ricettiva (RAM) ad ogni informazione, purché numerica, si intende.
Occorre quindi conoscere le RAM e le ROM e il procedimento che il tuo VIC 20 segue per ricordare (scrivere o leggere) una informazione.
Ma come penetrare nella memoria? Il tuo computer ha una risposta anche a questo: POKE e PEEK.
Esistono poi inoltre funzioni matematiche e non. In questa lezione ne analizzeremo alcune (INT, SQR, ABS) per finire con i contatori e i cicli controllati.

La memoria

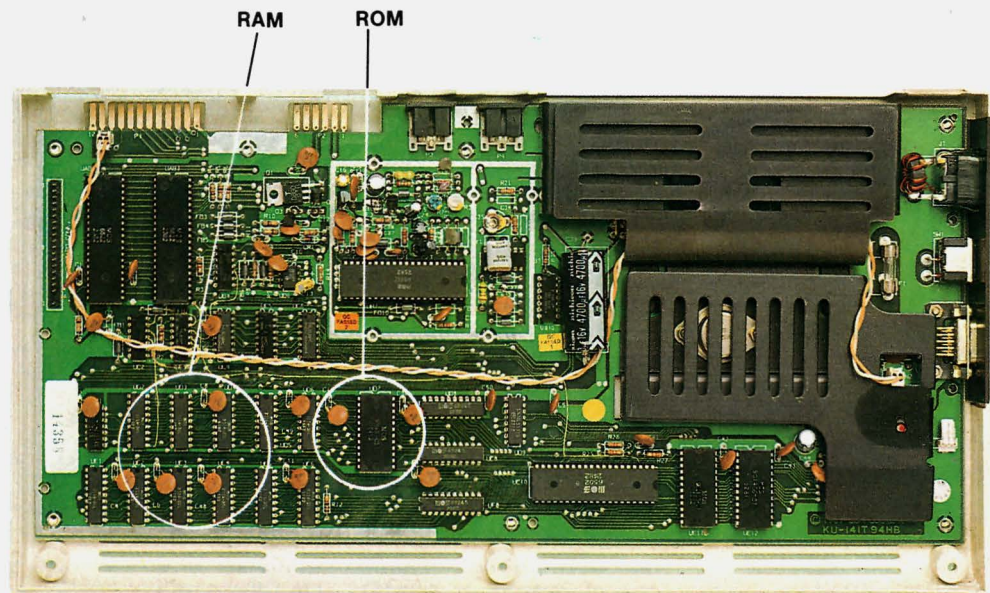
Abbiamo visto nelle scorse lezioni come il microprocessore svolga, all'interno del tuo VIC 20, l'indispensabile compito di elaborare le istruzioni e le *informazioni*.
Ciò che oggi ci proponiamo di capire è in quale modo esso possa, una volta portate a termine le singole operazioni, ricordarsi e tenere conto delle cose già fatte e di quelle ancora da fare.
Il problema, in realtà, è abbastanza semplice: non essendo dotata di *proprie ed autonome* capacità di memorizzazione, l'unità centrale (o CPU) sfrutta una memoria ausiliaria, alla quale si collega tutte le volte che desidera leggere o scrivere dei dati. In questo modo, nonostante le modeste ed elementari facoltà di calcolo di cui dispone, essa può eseguire, grazie ad *un'incredibile* velocità, operazioni che nel complesso risultano estremamente laboriose.
È quindi possibile dire che in un elaboratore la memoria rappresenta una sorta di magazzino, nel quale vengono depositate o prelevate tutte quelle informazioni

HARDWARE

che il processore ritiene essere importanti od utili ai fini dell'esecuzione di un programma.
(Tieni presente che tutte le informazioni sono espresse sotto forma di numeri, quindi le memorie non sono altro che magazzini, più o meno grandi, di numeri).
Fisicamente puoi immaginare la memoria di un computer come se fosse un microscopico alveare: in ciascuna celletta (o locazione) di questo alveare la CPU è in grado di leggere o scrivere un solo dato alla volta.
Analogamente, ogni locazione della memoria può conservare, stipata al proprio interno, un'unica informazione.
L'unità centrale individua le singole cellette mediante un indirizzo numerico, al quale fa riferimento quando desidera memorizzare o ritrovare un qualsiasi dato.
Il contenuto di una locazione della memoria corrispondente ad un certo indirizzo non è mai una celletta completamente vuota: è più o meno come se in ciascuna locazione fosse contenuto un lampadario (byte) a 8 lampadine (bit).
Così, quando la CPU desidera modificare ad un nuovo valore (per esempio a 150 decimale cioè 10010110 binario) il numero rappresentato dal lampadario posto in una certa celletta (ad esempio la 1533), è sufficiente che essa accenda o spenga le lampadine relative a quella locazione, fintanto che quel lampadario non indichi il nuovo valore (150).
Evidentemente, il numero rappresentato precedentemente dal lampadario della cella 1533 va perso e viene sostituito dal nuovo valore di 150 (10010110 in binario).
In maniera analoga il processore, quando desidera leggere il contenuto di un certo indirizzo della memoria, non fa altro che ricopiarsi la combinazione di lampadine accese o spente corrispondenti al lampadario della celletta posta a quell'indirizzo.
La cosa importante che ti devi quindi ricordare sulla memoria dei computer è che essa è individuata da indirizzi e contiene dei valori. Gli indirizzi indicano una specifica celletta (o locazione) della memoria. Ogni locazione contiene un solo valore alla volta. Mentre l'indirizzo di una locazione resta sempre uguale, il valore memorizzato in un indirizzo, invece, a volte può essere modificato a piacimento.
Come era però naturale attendersi, visto che non era materialmente possibile che il tuo VIC 20 potesse contenere al proprio interno migliaia

di lampadari, i progettisti della casa costruttrice hanno dovuto risolvere il problema del mantenimento delle informazioni ricorrendo ad altri mezzi.
Fermiamoci un attimo. Come hai visto, il concetto di "ricordo" per il tuo VIC 20 è alquanto semplice; non altrettanto si può dire del problema tecnico relativo.
L'elettrotecnica pura e l'elettronica lo hanno risolto proponendo non solo soluzioni sempre più efficienti, ma anche dispositivi di memoria diversi per proprietà e possibilità.

RAM e ROM

Benché, come ribadito, tutta la memorizzazione avvenga sotto forma di bit (in parole povere, attraverso opportune sequenze di interruttori accesi/spenti), esistono sostanziali differenze anche tra memoria e memoria.
Le due grandi famiglie sono:
— le ROM (Read Only Memory - Memorie di sola lettura), che sono permanenti: una volta impressionate sono immutabili, proprio come una pellicola fotografica;
— le RAM (Random Access Memory - Memorie ad accesso casuale), che sono memorie non permanenti, dette anche volatili, in cui l'informazione può essere cambiata a piacimento.
La ROM è come il motore di un'automobile e la RAM il suo serbatoio; basta che manchi uno solo dei due perchè la vettura sia inutilizzabile.
Senza la ROM il tuo VIC 20 (e la CPU) non sarebbe capace di svolgere alcun compito: non esisterebbe nessuno che gli ricorda in maniera permanente quanto deve fare. Senza la RAM non saprebbe dove mettere il risultato delle sue elaborazioni, pur essendo in grado di trattare dei dati.
Nella versione base del tuo VIC 20, le cellette che costituiscono la memoria sono più di 26.000 (tramite le varie espansioni possono però diventare ben 65536): ad ognuna di esse corrisponde un indirizzo, un numero, cioè, compreso tra 0 e 65535.
Ciascuna locazione può contenere un byte da 8 bit: il valore stipato in ogni singola celletta è quindi un numero compreso tra 0 e 255 (quest'ultimo è infatti il valore massimo rappresentabile da un numero binario ad 8 bit).
La memoria RAM serve per contenere i programmi ed i dati che vengono di volta in volta scritti ed usati dall'elaboratore quando gli chiedi di eseguire qualche operazione.
Essa è chiamata "ad accesso casuale", in quanto è possibile accedere ad una specifica locazione con la stessa facilità con cui è possibile accedere a qualsiasi altra.
Un ultimo, ma

significativo fatto sulla memoria.RAM è il seguente: quando spegni il computer tutto quello *che vi era contenuto* va definitivamente cancellato e dimenticato. Proprio per questa ragione si ricorre allora a dispositivi di memorizzazione periferica (ossia non facenti parte dell'elaboratore vero e proprio), come i registratori: essi possono infatti conservare sui nastri magnetici le informazioni ed i programmi che altrimenti andrebbero perduti.
Nella memoria ROM, invece, i dati ed i programmi vengono registrati direttamente e permanentemente dalla casa costruttrice. In essa sono contenute tutte quelle informazioni che il microprocessore pretende di conoscere all'atto dell'accensione del computer o al momento dell'esecuzione di un programma. La ROM è quindi tipicamente utilizzata per memorizzare quei programmi e quelle informazioni che non debbono subire nel tempo alcuna variazione.
L'interprete BASIC, per esempio, risiede stabilmente in una memoria ROM: in tal modo esso è subito pronto, non appena accendi il tuo VIC 20, a svolgere con diligenza il proprio lavoro.
Nel tuo VIC 20 esistono 16384 (corrispondenti a 16 K) cellette adibite a

memoria ROM, cioè riservate a quei dati ed a quei programmi indispensabili per il funzionamento del sistema.
Ricorda quindi: la ROM e la RAM costituiscono la memoria complessiva su cui agisce il VIC 20; la prima contiene quei programmi che devono mettere il computer in condizione di funzionare al momento dell'accensione e che quindi non possono essere perduti quando spegni il calcolatore. La RAM, invece, contiene i programmi ed i dati che di volta in volta scrivi ed utilizzi e che "abbandonano" la memoria quando smettono di essere utili e necessari: sta a te giudicare quando vale la pena di evitare che questi ultimi siano irrimediabilmente perduti, provvedendo eventualmente a salvarli su nastro o su disco, prima di staccare l'alimentazione al tuo computer.

Come "ricorda" un computer

La nostra memoria ricorda gli avvenimenti che più ci hanno colpito, quelli cioè che hanno lasciato una traccia. In un computer la sostanza è la stessa: per memorizzare qualcosa occcorre produrre una traccia, un cambiamento. Immagina una scacchiera con i pezzi in una posizione qualunque, come se avessi interrotto una partita a metà. La scacchiera è in un certo senso una memoria: mantiene l'informazione (la posizione dei pezzi) fino a nuovo ordine. Trasformiamo il tutto in termini elettronici. L'informazione (1 o 0) contenuta nel circuito (la scacchiera) è mantenuta in ciascuna casella fino a che non interviene una variazione (spostamento di un pezzo), che ne modifica lo stato (da 1 a 0 e viceversa). Questo è il principio generale su cui si basano tutte le RAM. Nelle ROM l'unica differenza è che la fase di cambiamento è stata inibita. Vediamo come si raggiunge lo scopo.
Una delle prime RAM costruite era a nuclei di ferrite, piccoli anelli di materiale ferromagnetico (ferrite appunto), pari al numero di informazioni da memorizzare. Un anello di questo tipo si può magnetizzare (diventa una calamita per intenderci) in modi differenti, a seconda del verso della corrente che lo attraversa. Senza dilungarci nella teoria dell'elettromagnetismo, ti basti sapere che:

- a ciascuno stato di memorizzazione viene associato un valore binario (0 o 1);
- vi sono fili verticali e orizzontali (matrice) attraverso cui si può, inviando corrente, modificare lo stato di un nucleo. Ciascun nucleo, infatti, è posto all'intersezione di un filo verticale con uno orizzontale. È un po' come giocare a battaglia navale: dando le coordinate (un filo orizzontale e uno verticale) si individua esattamente una casella (nucleo) di cui si può cambiare stato (0 o 1);
- vi è un filo di lettura che, attraverso tutti gli anelli della matrice, "legge" l'informazione senza alterarla.

Si tratta di circuiti concettualmente molto

semplici, la cui realizzazione, però, comportava non pochi problemi sia di natura economica (alto costo) che di spazio. Per darti un'idea, con una memoria a nuclei per memorizzare 4096 byte occorre uno spazio pari ad un cubo di 20 cm di lato. Due erano quindi le esigenze: comprimere la maggior quantità possibile di informazioni nel minor spazio e ridurre i costi.
Nacquero così dispositivi a semiconduttore: prima i transistor e poi i circuiti integrati (chip).
Ciascun chip è costituito da una minuscola piastrina di silicio sulla quale sono stati posti - con particolari e sofisticati accorgimenti - tutti quei dispositivi elettronici (diodi, transistor, resistenze, condensatori, ecc.) necessari al funzionamento del circuito. I pezzi di plastica nera con molti piedini (pin) in metallo, che puoi vedere sollevando il coperchio del tuo VIC 20, non sono altro che i contenitori di tali chip.
A proposito delle memorie a semiconduttore occorre fare una distinzione:

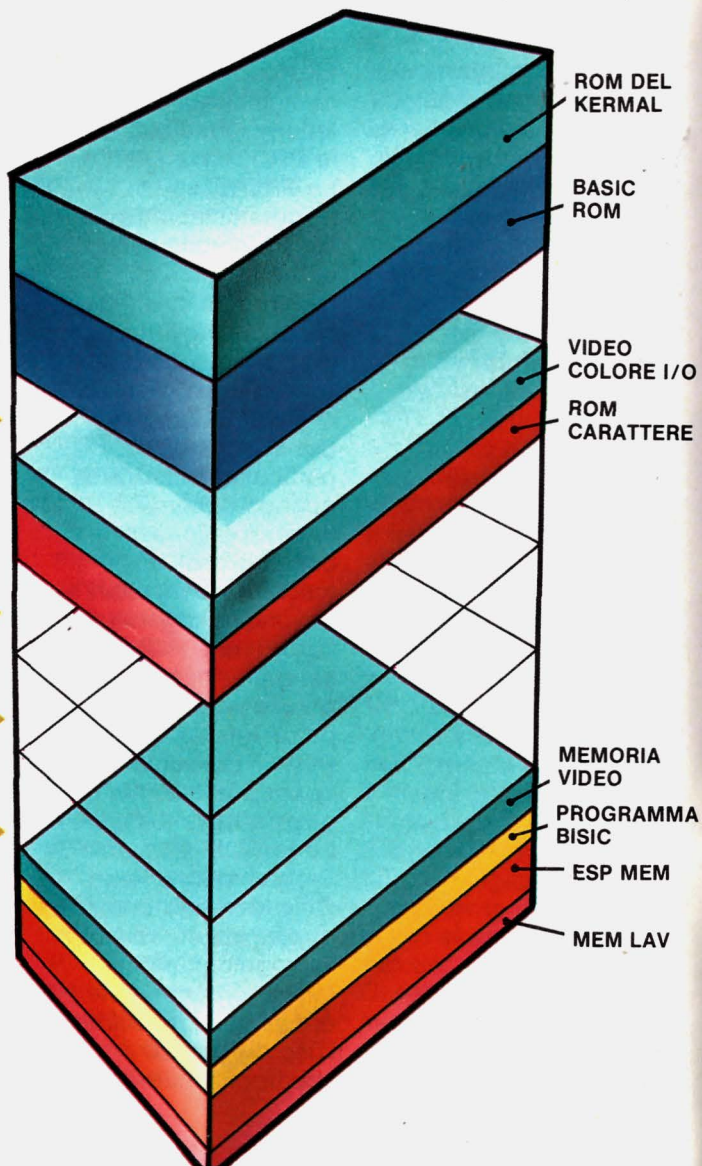
— memorie statiche: ogni bit è memorizzato da un circuito bistabile (che può assumere due stati stabili) e rimane senza problemi fino all'eventuale caduta (mancanza) della tensione di alimentazione;
— memorie dinamiche: i bit sono memorizzati in microscopici condensatori, che si scaricano lentamente e richiedono un continuo refresh (rinfresco) dell'informazione.

Le informazioni contenute in memoria vengono ripetute periodicamente (cioè la memoria viene impressionata più volte), in modo da non perderle.
Nei primi computer questo lavoro veniva svolto dalla CPU che, ad intervalli di tempo determinati (inferiori al millesimo di secondo), ricopiava tutto il contenuto delle memorie. A prima vista può sembrare che la CPU debba svolgere troppi compiti, tuttavia il memory refresh non pregiudica l'attività della CPU più di quanto il singhiozzo non infastidisca un essere umano. A tratti si interrompono le normali attività, ma tra un singhiozzo e l'altro si è liberi di fare ciò che si vuole.
E il rinfresco, per una CPU che ha una frequenza di clock nell'ordine del MHz, non è niente di più che un singhiozzo. Un MHz (Megahertz) vuol dire che l'orologio del computer fa un milione di giri al secondo. Veloce, vero?!
Quest'ultimo tipo di memoria RAM è quello usato nel tuo VIC 20 e in genere in tutti gli altri

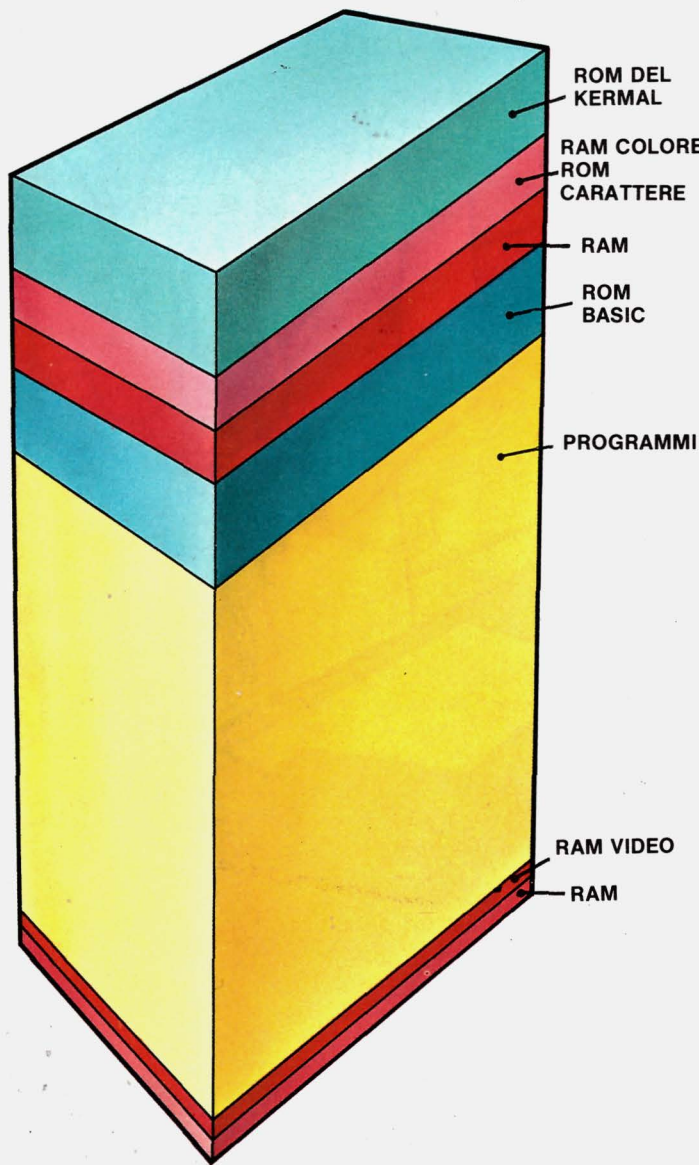
personal computer.
Vediamo ora le ROM. Sono memorie non volatili, che, come detto, mantengono sempre l'informazione. Sono in sostanza, formate da una maglia di semiconduttori, cioè una serie di conduttori organizzata in righe e colonne. Ogni incrocio tra righe e colonne rappresenta un bit a livello logico 0 o 1, a seconda che riga e colonna siano unite o meno tra loro. La programmazione (cioè la realizzazione della maglia) viene fatta in base alle esigenze di chi la fabbrica, ed una volta stabilita è definitiva (non può essere modificata in alcun modo).
Esistono anche altri tipi di memoria a sola lettura: le più usate sono le PROM e le EPROM.
Le PROM (Programmable Read Only Memory) sono, come dice il nome, memorie di sola lettura, programmabili dall'utente grazie a dispositivi speciali (i programmatori di PROM). Vengono usate per scopi particolari, soprattutto quando occorrono ROM in pezzi unici per collaudi, prototipi, macchine particolari e dovunque in generale l'uso di una ROM sarebbe troppo costoso, perché applicato a pochi pezzi. Infatti l'uso di ROM è conveniente quando la quantità di pezzi supera il migliaio.
Le EPROM (Erasable Programmable Read Only Memory), anch'esse di sola lettura, hanno il vantaggio di poter essere programmate più di una volta mediante l'uso di speciali apparecchiature.

Le mappe di memoria

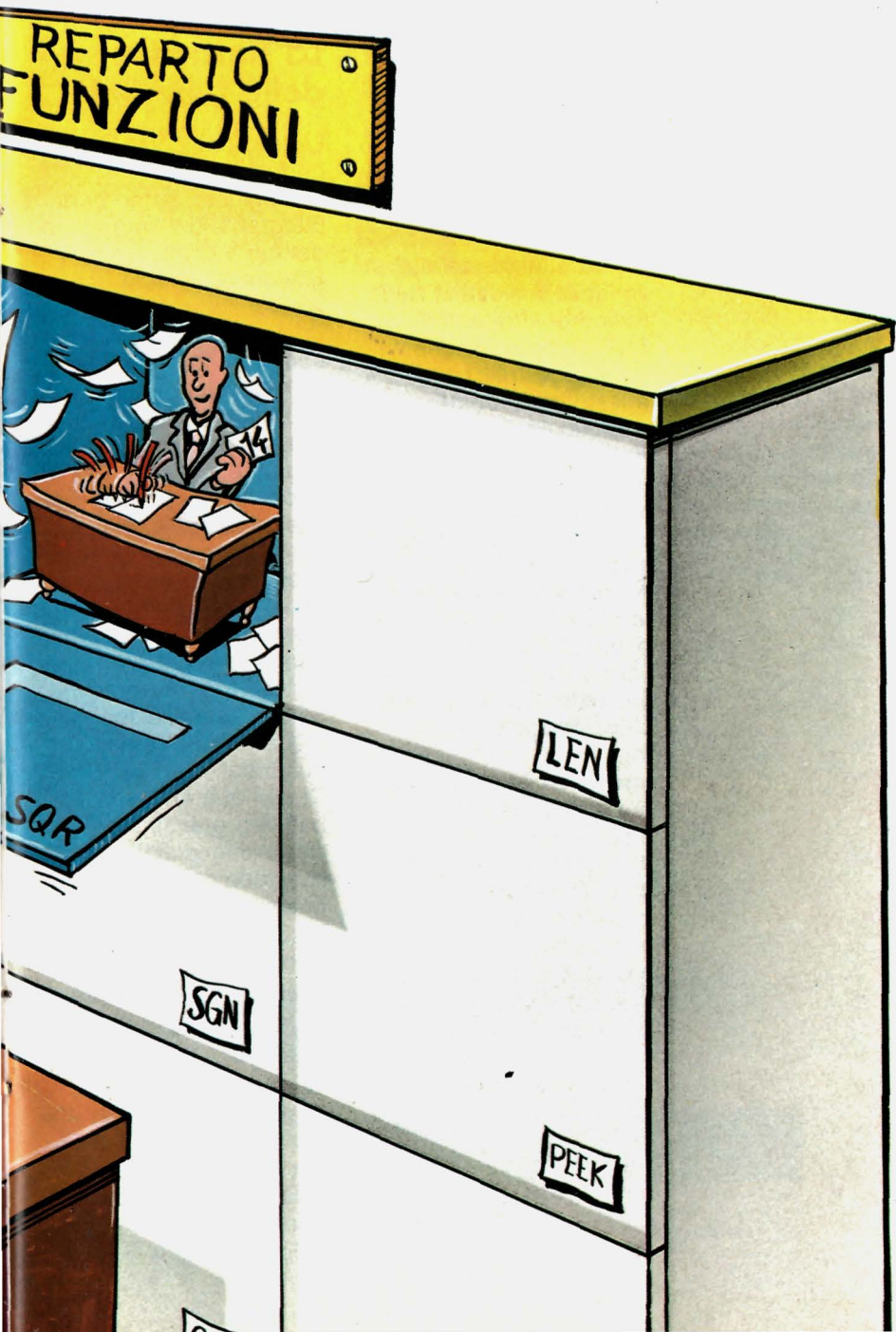
Fino ad ora abbiamo spiegato in che modo e con quali sistemi si memorizza un'informazione. Ma in che modo la si legge? E, soprattutto, come sapere che informazione devo prendere e dove?
Immagina di trovarti in una grande biblioteca, fornita di decine di migliaia di volumi. Come organizzare l'archivio in modo da poter trovare rapidamente un libro?
Una soluzione accettabile è ordinare per numero progressivo i volumi, in modo da sveltire al massimo la ricerca di un libro.
In un computer la soluzione è analoga: ogni cella di memoria (quella zona, cioè, che contiene una "parola" o byte) è identificata da un numero (l'indirizzo).
Normalmente la quantità di memoria che un computer deve tenere sotto controllo è di circa 64 Kbyte cioè 65536 byte (in decimale, il simbolo K vale mille, ma in binario 1024, cioè 2^{10}).
Indirizzare significa dare un numero, un indirizzo ad ogni cella di memoria (numero progressivo dei

volumi), definito *all'interno del computer* in maniera unica e costante. Inoltre, vista l'eterogeneità dei dati che il computer deve trattare (dei libri per il nostro bibliotecario), chi progetta la macchina li divide in sezioni (materie).
Viene così creata la cosiddetta mappa di memoria, cioè *un'organizzazione dei* dati in essa contenuta, fatta in modo da raggruppare tutti quei byte che hanno qualcosa in comune.
La mappa della memoria non è altro che una dettagliata spiegazione, sulla presenza e sulla disposizione delle varie locazioni adibite ai diversi compiti, che la casa costruttrice fornisce insieme al calcolatore.
Come puoi vedere sul manuale del tuo VIC 20. Essendo questi compiti

I disegni qui a fianco rappresentano le mappe di memoria del C 64 (a destra) e del VIC 20, cioè dove sono o vengono memorizzate e organizzate le informazioni.

molteplici e talvolta complessi, la mappa di memoria del tuo VIC 20 potrebbe sembrarti alquanto difficile.
Non preoccuparti però: essa esiste per permetterti di ignorarla! Per rendere cioè possibile lo svolgimento delle varie funzioni del computer, senza che tu debba preoccupartene.
Gli elementi più importanti della mappa di memoria del VIC 20 (versione base) sono:

— La ROM del BASIC (8 k) posta tra gli indirizzi 49152 e 57343 contenente l'interprete BASIC.
— La ROM Kernal (8 k tra 57344 e 45535) dove risiedi il sistema operativo.
— La ROM caratteri (4 k tra 32768 e 36863) che contiene i dati dei simboli che puoi visualizzare (lettere, cifre, caratteri grafici, ecc.).
— La RAM video (0,5 k tra 7680 e 8191) dove vengono memorizzati i caratteri che operano sullo schermo.
— La area dei programmi BASIC (3,5 k tra 4096 e 7679) dove vengono memorizzati i programmi BASIC.

Le funzioni

Prova per un attimo a prendere in mano la tua calcolatrice tascabile. Molto probabilmente essa è in grado di risolvere, oltre alle normali operazioni aritmetiche, anche calcoli più complicati ed impegnativi delle semplici sottrazioni o moltiplicazioni: per esempio, l'estrazione di radici quadrate o l'elevamento a potenza di un numero.
In BASIC esistono alcune espressioni che ti consentono di ottenere dal VIC 20 lo stesso risultato di una calcolatrice: le funzioni matematiche.

REPARTO FUNZIONI
14
LEN
SQR
SGN
PEEK

Una funzione è un operatore al quale fornisci una o più variabili indipendenti e che esegue su di essa delle elaborazioni per fornire quell'unico valore finale che costituisce il risultato (variabile dipendente). Si scrive in termini generali: y = f (x), dove x è la variabile indipendente e y quella dipendente. Per esempio:

```
LET A = SQR (8)
```

significa: assegna alla variabile A il valore della radice quadrata di 8. SQR ( ) è la funzione che è stata utilizzata in questo caso: come risultato essa fornisce la radice quadrata dell'espressione (chiamata anche argomento) racchiusa entro le parentesi. Argomenti diversi forniscono ovviamente risultati diversi. Se l'argomento non è valido, la funzione non viene eseguita ed *innesca la* visualizzazione di un messaggio di errore. Così, se nell'esempio precedente avessimo chiesto al computer di eseguire questa istruzione:

```
LET A = SQR (—8)
```

avremmo ottenuto come unico risultato un ILLEGAL QUANTITY ERROR, cioè quantità non legale: non è infatti possibile la radice quadrata di un numero negativo!

La sintassi delle funzioni

Le funzioni, come di certo hai già immaginato, fanno parte *integrante del linguaggio* del tuo VIC 20: esse risiedono infatti in una zona della memoria ROM alla quale il computer fa riferimento tutte le volte in cui intendi utilizzarne qualcuna. Per riferirti alle varie funzioni devi quindi adoperare la parola riservata che la ditta costruttrice ha assegnato a ciascuna di esse: normalmente è una abbreviazione della corrispondente parola inglese (per esempio: SQR è abbreviazione di SQuare Root, radice quadrata).
Qualunque errore di battitura o di sintassi viene immediatamente individuato.
In termini generali una funzione ha questo formato:

```
funzione (argomento)
```

Ogni argomento può essere una costante, una variabile od una espressione.
Una funzione presente in una istruzione viene calcolata prima di ogni

altro operatore.
L'argomento di una funzione può essere ancora una funzione.
Le funzioni non possono mai comparire a sinistra del segno di uguale. Una istruzione di questo tipo:

```
SQR (B) = A
```

costituisce un sicuro veicolo per provocare l'invio di un messaggio di errore da parte del tuo VIC 20.
Nel complesso le funzioni disponibili, e non solo matematiche, sul tuo VIC 20, sono circa una ventina: passo dopo passo imparerai a conoscerle tutte e a utilizzarle con facilità.

SQR

La prima di queste funzioni l'hai già conosciuta: è SQR (abbreviazione, come detto, di SQuare Root, radice quadrata), l'amica che ti restituisce la radice quadrata del numero che tu le avrai fornito come argomento.
Tutto ciò che essa pretende è che l'argomento su cui essa deve operare sia positivo, od al massimo nullo. In caso contrario l'interprete BASIC ti invia un segnale di errore ILLEGAL QUANTITY ERROR, arrestando l'esecuzione.
Supponiamo di voler calcolare e visualizzare sul video la radice quadrata di 3; l'istruzione dovrà pertanto essere:

```
PRINT SQR (3)
```

L'argomento all'interno delle parentesi può essere una qualsiasi espressione "legale" BASIC (cioè accettabile dall'interprete). Un comando di questo tipo:

```
PRINT SQR (3 * 4 — (SQR (16/4))),
```

per quanto possa non risultarti di immediata comprensione, è perfettamente ammissibile all'interno di un programma o di un'istruzione.
Il tuo VIC 20 è preciso, ma non preciso in assoluto: di un numero, stampa al massimo 9 cifre, mentre ne conserva e considera in memoria, 10. Qualsiasi metodo usi, infatti, per il calcolo della radice quadrata, o di altre operazioni complesse, il risultato sarà sempre una approssimazione del valore reale. Tienilo presente.

Esempi

Istruzione	Spiegazione
LET A = SQR (5.5 + 3.5)	Assegna alla variabile A il valore della radice quadrata di 5.5 + 3.5, cioè 3.
LET B = SQR (20 − SQR (16))	La radice quadrata di 16 è 4:20 meno 4 dà 16. Alla variabile B viene pertanto assegnato il valore 4.
LET C = SQR (NUMERO)	NUMERO è una variabile. Attento: in caso contrario questa istruzione provoca un errore!
LET D = 4 * SQR (9) − 2	Assegna a D il valore 10, ottenuto risolvendo la semplice espressione a destra dell'=. Come hai potuto vedere, se nella stessa espressione compaiono insieme funzioni ed operazioni aritmetiche, le funzioni sono le prime ad essere calcolate. La loro priorità di esecuzione è quindi massima. Come al solito, però, l'uso delle parentesi può modificare l'ordine di esecuzione dei calcoli.
LET NUM1 = SQR (25) − SQR (256)	Questa istruzione assegna a NUM1 il valore 5 − 16 = − 11.
LET NUM2 = SQR (25 − SQR(256))	Le parentesi hanno modificato l'ordine di priorità: l'istruzione assegna pertanto alla variabile NUM2 il valore della radice di 25 − 16, cioè 3.

INT

INT è un'altra funzione disponibile sul tuo computer. Sta a significare: parte intera dell'argomento. Essa fornisce cioè come risultato il numero intero (approssimato per difetto) più vicino all'argomento reale che tu hai racchiuso entro parentesi. Perciò:

```
INT (34.125) = 34
```

L'uso della funzione INT applicata ai numeri negativi con decimali fornisce il numero negativo intero immediatamente inferiore all'argomento:

```
INT (—4.41) = —5
```

Tieni quindi presente che mentre nel caso dei numeri positivi il risultato della funzione INT equivale al valore dell'argomento privato della parte decimale, quando l'argomento è un numero negativo devi pensare al valore intero immediatamente più piccolo dell'espressione di partenza.

Molti possono essere i possibili utilizzi della funzione INT: per esempio, le banche - quando a fine anno calcolano gli interessi maturati dai libretti di risparmio - arrotondano tutti gli importi proprio per mezzo di questa funzione.
Attraverso l'uso di INT, inoltre, puoi separare la parte intera e la parte decimale di un numero reale.
Infatti le istruzioni:

```
LET B = INT(A)
LET C = A - INT (A)
```

provocano l'assegnamento alle variabili B e C della parte intera e della parte decimale dell'argomento, costituito dalla variabile A.

Dall'esempio che segue puoi osservare come la funzione INT sembri proprio fatta apposta per le banche e non per i loro clienti ... Scherzi a parte, spero ti possa chiarire l'effetto di INT sui numeri decimali.

```
 10 REM PRIMO VANTAGGIO (SUI DEPOSITI)
 20 LET RF = 985000.9
 30 REM RF = RISPARMIO ALLA FINE DELL'ANNO
 40 LET RN = INT (RF): PRINT RN
 50 REM IL TUO RISPARMIO È CALATO DI ...
 60 REM SECONDO VANTAGGIO (SUI PRESTITI)
 70 LET DF = − 1450.1
 80 REM DF = DEBITO ALLA FINE DELL'ANNO
 90 LET DN = INT (DF): PRINT DN
100 REM IL TUO DEBITO È AUMENTATO DI ...
```

Esempi

```
LET X = 33
LET Y = 100
LET Z = INT (Y/X)
```

assegna alla variabile numerica Z la parte intera del rapporto tra i valori contenuti nelle variabili numeriche X e Y (Z quindi vale 3).

```
LET NUMERO = SQR (INT (100.12))
```

La variabile NUMERO assume il valore della radice di 100, cioè 10

```
LET A = 21.5
LET B = 22
LET C = INT (A − B)
```

Poiché il valore dell'argomento della funzione è un numero decimale negativo (infatti A − B risulta − 0.5), la variabile C assume il valore intero ad esso immediatamente inferiore (cioè − 1).

SGN

Molte volte può essere utile conoscere se una variabile (od un numero, oppure un'espressione) risulta positiva, negativa o nulla: alla soluzione di tale problema provvede la funzione SGN (dall'inglese sign), detta anche funzione segno. Qualunque sia il numero iniziale (intero o decimale), essa fornirà infatti come risultato:
1, se il valore dell'argomento è positivo;
0, se il valore dell'argomento è nullo;
—1, se il valore dell'argomento è negativo.
Ad esempio:

```
LET AA = SGN (33.25)
LET BB = SGN (4 – SQR (16))
LET CC = SGN (– 13.2)
```

Queste istruzioni assegnano rispettivamente alle variabili AA, BB e CC i valori 1, 0 e —1.
La funzione segno ricopre un ruolo singolare nella programmazione. Essa, a *differenza di altre, non* ha un vero e proprio effetto ma "segnala" all'interno del programma una determinata caratteristica (il segno appunto).
La sua funzione principale è proprio "avvertire" il programma del mutato segno di una espressione. Ne è un chiaro esempio il programma seguente:

```
10 REM VERIFICA FINANZIARIA CON SGN
20 INPUT "QUANTI SOLDI AVEVI IERI?"; SI
30 INPUT "QUANTI SOLDI HAI OGGI?; SO
40 LET DIFF = SI – SO
50 IF SGN (DIFF) = 0 THEN PRINT "NESSUNA VARIAZIONE"
60 IF SGN (DIFF) = – 1 THEN PRINT "HAI PIU' SOLDI"
70 IF SGN (DIFF) = 1 THEN PRINT "HAI FATTO SPESE"
```

ABS

ABS è un'altra funzione numerica. Il suo compito è quello di convertire il valore dell'argomento in un numero positivo, indipendentemente dal segno posseduto da quest'ultimo. ABS è infatti l'abbreviazione dell'inglese ABSolute Value (valore assoluto). Quindi, per esempio:

```
ABS (- 3.41) = ABS (3.41) = 3.41
```

L'utilizzo di questa funzione è particolarmente utile quando si vogliono effettuare dei calcoli senza doversi preoccupare dei vari segni. Così, per essere sicuri che una espressione di questo tipo:

```
LET C = SQR (TOTALE)
```

sia corretta in qualsiasi eventualità, è sufficiente scriverla come:

```
LET C = SQR (ABS (TOTALE))
```

Agendo in tal maniera il segno della variabile TOTALE non influenzerà più in modo così drastico l'esito dell'operazione, mettendoci al riparo da possibili errori ed arresti del programma.
È immediatamente intuibile come opera il tuo VIC 20 per ottenere la funzione ABS: basta mettere il codice del segno + nella locazione di memoria che contiene il segno del numero desiderato.
Il valore assoluto, infatti, considera i numeri solo come positivi.
Un ottimo esempio dell'uso cui questa funzione può essere

investita all'interno di un programma è il calcolo della lunghezza di un segmento nel piano.

Immaginiamo un piano cartesiano dove X1 e X2 sono gli estremi del segmento di cui dobbiamo determinare la lunghezza, la formula da usare è:

$$\sqrt{| X1 - X2 |^2 + | Y1 - Y2 |^2} = \text{LUNGHEZZA SEGMENTO}$$

dove le | | rappresentano matematicamente il valore assoluto dell'espressione contenuta all'interno. Ecco il programma:

```
10 REM CALCOLA LA LUNGHEZZA
   DI UN SEGMENTO NEL PIANO
20 INPUT "SCRIVI GLI ESTREMI DEL SEGMENTO
   LUNGO L'ASSE X"; X1, X2
30 INPUT "SCRIVI GLI ESTREMI DEL SEGMENTO
   LUNGO L'ASSE Y"; Y1, Y2
40 REM CALCOLO DELLA LUNGHEZZA
50 LET LUNGHEZZA = SQR (ABS (X1 - X2) ↑ 2 +
   (ABS (Y1- Y2) ↑ 2)
60 PRINT "LA LUNGHEZZA DEL SEGMENTO E";
   LUNGHEZZA
70 INPUT "VUOI CHE NE CALCOLI ANCORA?
   (S/N)"; A$
80 IF A$ ="S" THEN GOTO 20
90 END
```

π

Il π (pi greco) non è una funzione vera e propria: è piuttosto un valore costante che con tale nome è stato permanentemente memorizzato all'interno del tuo VIC 20.
Il suo scopo è molto semplice: evitare che tutte le volte in cui questo numero si rende necessario tu sia costretto a battere una per una tutte le varie cifre che lo compongono. Per vederne l'effetto è sufficiente che tu batta:

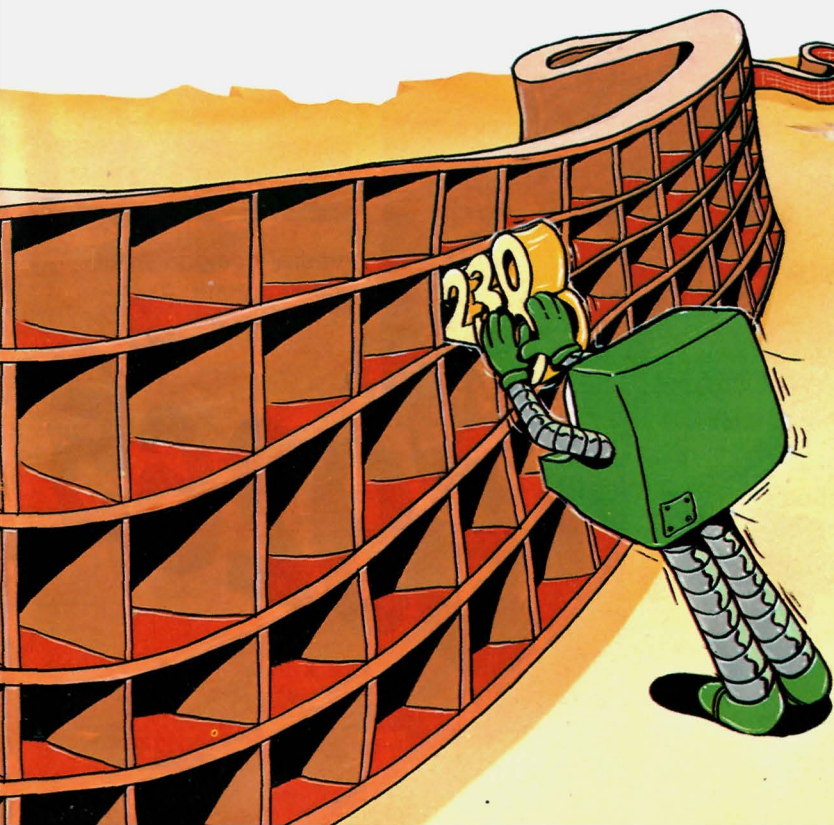
```
PRINT π
```

Vedrai comparire sul video il numero 3.14159265.

POKE

All'interno del tuo VIC 20, in ciascuna delle migliaia di locazioni che compongono la memoria, ogni informazione è memorizzata sotto forma di byte, cioè mediante sequenze di numeri compresi tra 0 e 255. L'interprete BASIC ti consente di curiosare o modificare i singoli valori presenti in ciascuna delle cellette mediante PEEK e POKE. Vediamo per prima POKE (che tradotto in italiano significa "mettere il naso in"). POKE serve per memorizzare un numero direttamente in una posizione della memoria, scavalcando i normali meccanismi del BASIC. Essa ti consente infatti di scrivere in una certa locazione della memoria qualsiasi numero compreso tra 0 e 255. Per esempio:

```
POKE 7650, 100
```

seguito da return

avrà come effetto di inserire nella locazione della memoria avente indirizzo numero 7650 il valore dell'argomento, cioè 100. POKE, non è una funzione, perciò non richiede l'uso delle parentesi. Il primo valore indica la cella di memoria, il secondo il dato da memorizzare.
Il valore che desideri scrivere nella memoria deve essere maggiore di zero e minore di 255: in caso contrario il VIC 20 si arresta, inviando un messaggio per avvertirti dell'errore.

```
ILLEGAL
QUANTITY ERROR
```

Il comando POKE ha naturalmente effetto soltanto sulle locazioni appartenenti alla memoria RAM: se l'indirizzo si riferisce infatti ad una cella della *memoria ROM, nulla* accade in seguito all'esecuzione di una POKE. Ricordati: la memoria ROM non è *modificabile!*
Non temere, quindi: con un comando POKE (come d'altra parte con qualsiasi altro comando BASIC) è assolutamente impossibile rovinare in maniera definitiva ed irrimediabilmente la memoria del tuo VIC 20!
Sia l'indirizzo che il dato possono essere forniti *sotto forma di* espressione o di variabile numerica. Ad esempio:

```
POKE 2000 * 3 + 1650, SQR (10000)
```

Come anche

```
LET I = 7650 : LET D = 100 : POKE I, D
```

In generale tutti i dati che il computer ha in RAM possono essere modificati a seconda delle esigenze di chi programma. Importante è sapere quali bisogna modificare per ottenere il risultato desiderato.

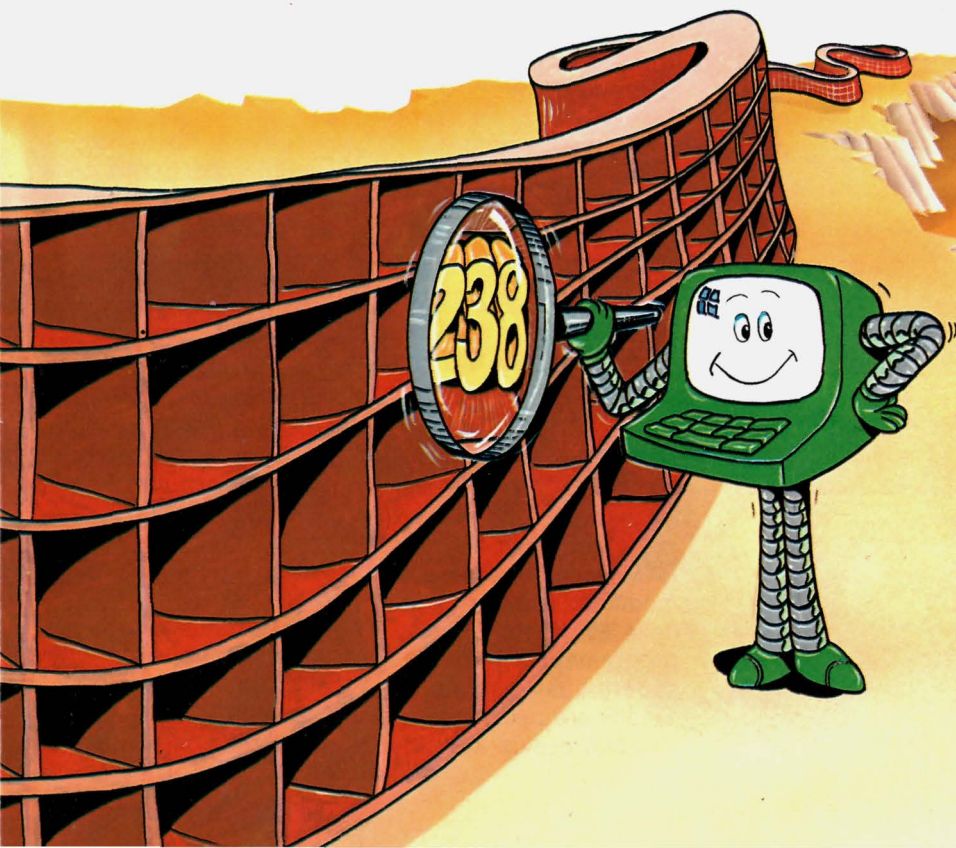
Queste informazioni sono contenute nelle variabili di sistema, numeri usati dal computer per "ricordarsi" di determinate operazioni. *Ad esempio, nel tuo* VIC 20 c'è una cella di memoria RAM in cui il sistema operativo, al momento dell'accensione, va a scrivere un numero per avvertire il computer che solo la barra spaziatrice ha, per così dire, il permesso dell'autorepeat.
Con ogni altro tasto, infatti, per ottenere nuovamente il carattere corrispondente, è necessario ribatterlo.
L'indirizzo di quella cella è 650 ed il valore scritto dal sistema operativo cioè la variabile di sistema è 0: in altre parole quando l'indirizzo 650 contiene 0, solo la barra spaziatrice ha l'autorepeat.
Grazie a POKE, però, tu puoi ora modificare quel valore: prova con POKE *650, 255 poi tieni* premuto il tasto della lettera o della cifra che preferisci ...
Visto il risultato?
Inserisci di nuovo: POKE 650, oppure spegni poi riaccendi (dopo 10 secondi) il computer e

tutto sarà come prima. Puoi dunque modificare le variabili del sistema. Attento, però, a non cambiare alcune variabili "chiave" essenziali al computer, poiché altrimenti causerai un arresto totale del sistema che, per un comando improprio (la variabile di sistema alterata), non sa più come continuare a lavorare e si blocca in gergo si dice "va in crash" (incidente).

PEEK

PEEK ("sbirciare"), è in un certo senso l'esatto contrario di POKE: essa ti permette infatti di conoscere il valore contenuto in una qualsiasi locazione della memoria. Così, se in precedenza hai eseguito l'istruzione POKE 7650, 100, il risultato del comando:

```
PRINT PEEK (7650)
```

sarà quello di stampare sullo schermo il numero 100: la PEEK non avrà fatto altro che leggere il contenuto della locazione numero 7650, mentre PRINT ne visualizzerà il valore sul video.
La funzione PEEK può essere applicata sia alla memoria RAM che alla

memoria ROM: ciò è facilmente comprensibile, dal momento che l'operazione di lettura è eseguibile su qualsiasi locazione della memoria. L'argomento di PEEK, cioè l'indirizzo della locazione da cui leggere il valore, deve sempre essere un numero compreso tra 0 ed il limite massimo della memoria, cioè 65535; un indirizzo al di fuori di questi limiti provoca l'arresto del programma e la visualizzazione di un messaggio di errore.

ILLEGAL
QUANTITY ERROR

A prima vista l'uso di PEEK e POKE potrebbe sembrare superfluo: quando mai ti potrà interessare essere in grado di scrivere o di leggere in una celletta? Come hai già letto, e te lo ripeto, essi invece sono comandi estremamente utili e potenti. La memoria, infatti, possiede alcuni magici indirizzi: riferendoti ad essi mediante PEEK o POKE puoi ottenere, per esempio, effetti grafici e sonori assolutamente impensabili con altri comandi BASIC.
Dall'altra parte della medaglia, però, scrivendo per errore un numero sbagliato in un indirizzo sbagliato hai la possibilità di perdere in un attimo programmi ed informazioni, che magari hanno richiesto ore di battitura sulla tastiera.
Morale: se non sei più che sicuro di ciò che stai facendo, cerca di evitare - soprattutto all'interno di lunghi programmi - l'uso di PEEK e, in particolar modo, di POKE: dovrai magari rinunciare a qualche possibile effetto coreografico, ma avrai la certezza di non avere inserito alcun comando che possa riservarti cattive sorprese e risultati indesiderati.
Concentrati e rifletti sulle analogie e sulle differenze tra

```
LET A = 37 : PRINT A
POKE 7660, 37 : PRINT PEEK (7660)
```

FRE

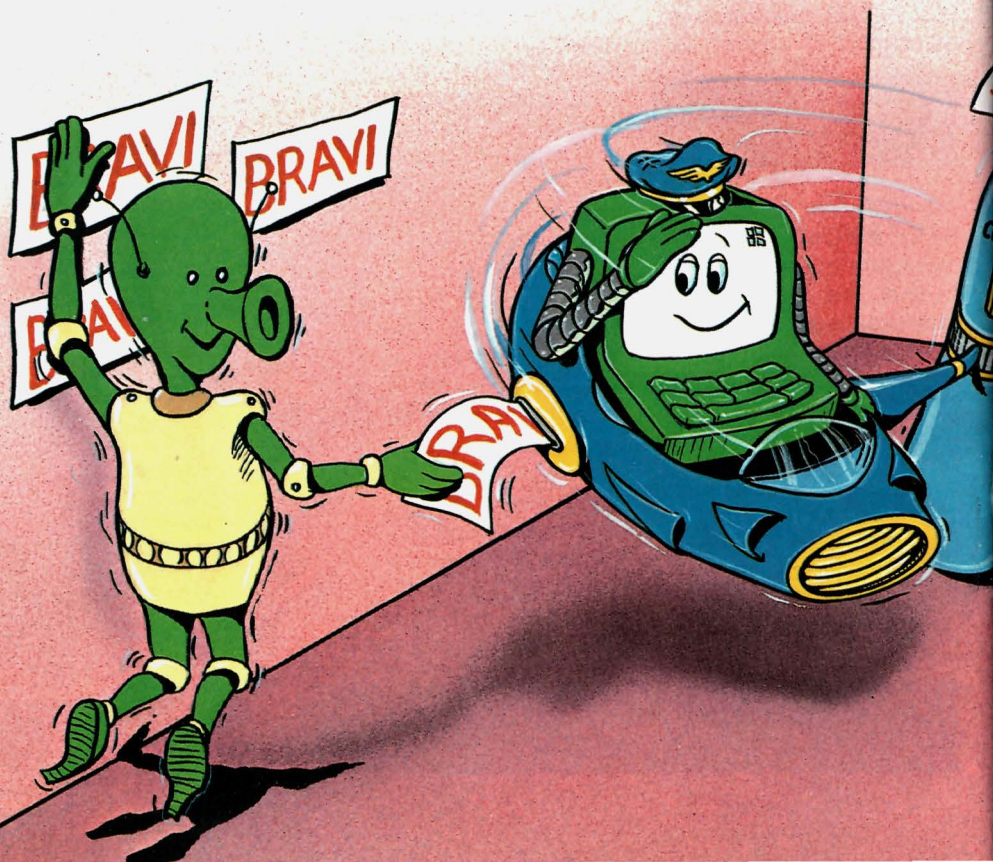
Fre deriva dal termine inglese free (libero); è una funzione, infatti, che ha come compito quello di comunicarti il numero di byte di memoria liberi (disponibili) al momento della sua esecuzione.
Va ricordato che se il numero ritornato è negativo, occorre aggiungere 65536 per ottenere il risultato corretto.
L'argomento della funzione FRE è del tutto fittizio. Pertanto sia PRINT FRE (0) che PRINT FRE (X) sono LEGALI.

Il contatore ed i cicli controllati

Esistono molti casi e situazioni nelle quali un elaboratore è chiamato ad eseguire, per un certo numero di volte, la stessa operazione.

Supponiamo, per esempio, di voler stampare tutti i numeri interi compresi tra 1 e 50. Un possibile programma potrebbe essere:

```
10 PRINT 1
20 PRINT 2
30 PRINT 3
40 PRINT 4
```

e così via.
Ti rendi subito conto che questo modo di procedere è inefficace ed inammissibile: la fatica di dover scrivere 50 istruzioni per risolvere un compito così banale sarebbe sicuramente eccessiva. Fortunatamente, però, esiste un modo più semplice e razionale col quale affrontare il problema: utilizzare un contatore.
Il contatore non è altro che una variabile adibita al conteggio di una certa sequenza di istruzioni/

azioni eseguite dal computer. Per mezzo di questa variabile è così possibile controllare il numero di volte in cui tali azioni/istruzioni vengono portate a compimento.
Nel nostro caso il programma potrebbe essere scritto in questo modo:

```
10 LET NUM = 1
```

Questa linea definisce la variabile NUM (il contatore) e ne pone il valore a 1

```
20 PRINT NUM
```

La linea 20 stampa il contatore

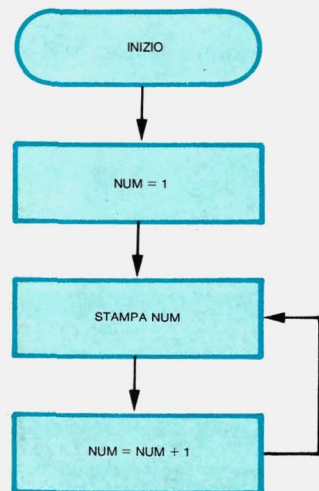
```
30 LET NUM = NUM + 1
```

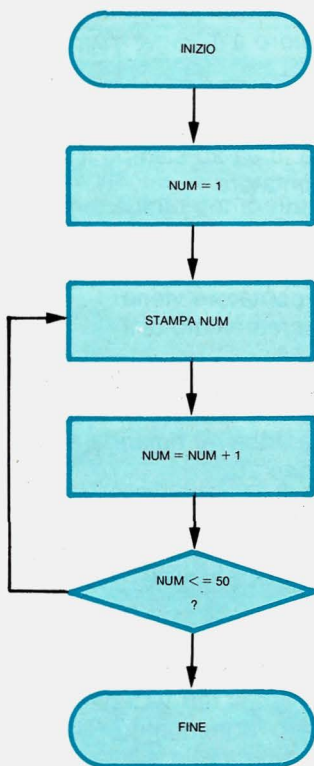
Il contatore viene incrementato di 1

```
40 GOTO 20
```

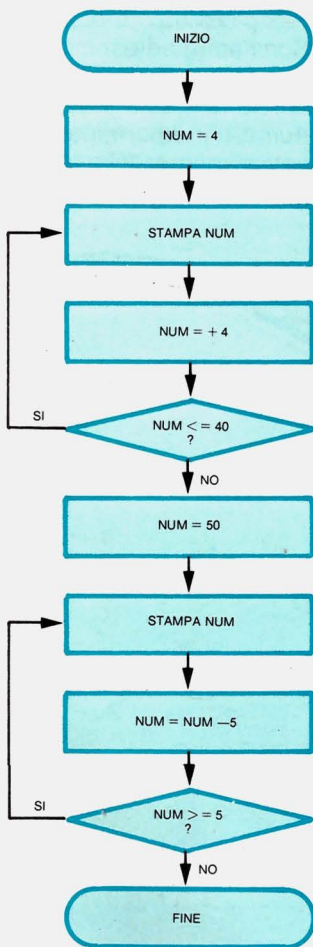
La linea 40 rimanda alla linea 20.
Ma che uguaglianza è mai:

```
NUM = NUM + 1
```

Vista come una normale equazione non ha alcun senso. Il tuo VIC 20, però, non la interpreta in questo modo, ma come un'assegnazione: prima calcola l'espressione a destra dell'uguale poi pone il risultato nella variabile NUM.
Prova a battere queste istruzioni ed a impartire il RUN. Comodo, vero? Con sole quattro istruzioni hai risolto il problema. Quando ti sarai stancato di veder girare i numeri sullo schermo batti RUN/STOP per far terminare lo spettacolo; il tuo VIC 20 ti avviserà di essersi fermato, visualizzando un messaggio di questo genere:
BREAK IN LINE 20
Ti sarai sicuramente accorto di come il nostro programma non soddisfi esattamente l'obiettivo che ci eravamo proposti: esso infatti prosegue nel conteggio e nella stampa dei numeri ben oltre il traguardo dei fatidici 50. La ragione è abbastanza semplice da individuare: cerchiamo di rappresentare il programma in termini di schema a blocchi:

PROGRAMMAZIONE

Il tuo VIC 20, eseguendo il precedente programma entra (come si dice nel gergo informatico) in un ciclo infinito, cioè in una sequenza di istruzioni dalle quali non ha modo di uscire.
L'errore che abbiamo commesso è uno dei più frequenti nei quali di solito incorre il programmatore alle prime armi: dimenticarsi di inserire una condizione da soddisfare perché il programma possa continuare.
L'unico rimedio a questo errore è quindi costituito dall'inserimento di un nuovo blocco, all'interno del quale eseguire un confronto tra il contatore ed il limite massimo (in questo caso uguale a 50).
La situazione diventa pertanto:

alla quale corrisponde il seguente programma BASIC:

```
10 LET NUM = 1
20 PRINT NUM
30 LET NUM = NUM + 1
40 IF NUM < = 50 THEN GOTO 20
50 END
```

La linea 40 corrisponde al "blocco delle decisioni".
Naturalmente non è affatto obbligatorio che il contatore venga incrementato con un valore unitario: al limite può anche essere decrementato (purché naturalmente la condizione venga adeguata alla nuova circostanza).
Quello che segue è un esempio di tale situazione: scopo del programma è infatti quello di stampare in ordine crescente la tabella pitagorica del 4 e, in ordine decrescente, quella del 5:

al contatore viene assegnato, come valore di partenza, il valore 4

l'incremento del contatore è 4

quando il programma è arrivato in questo punto la prima tabellina (quella del 4) sarà stata già stampata

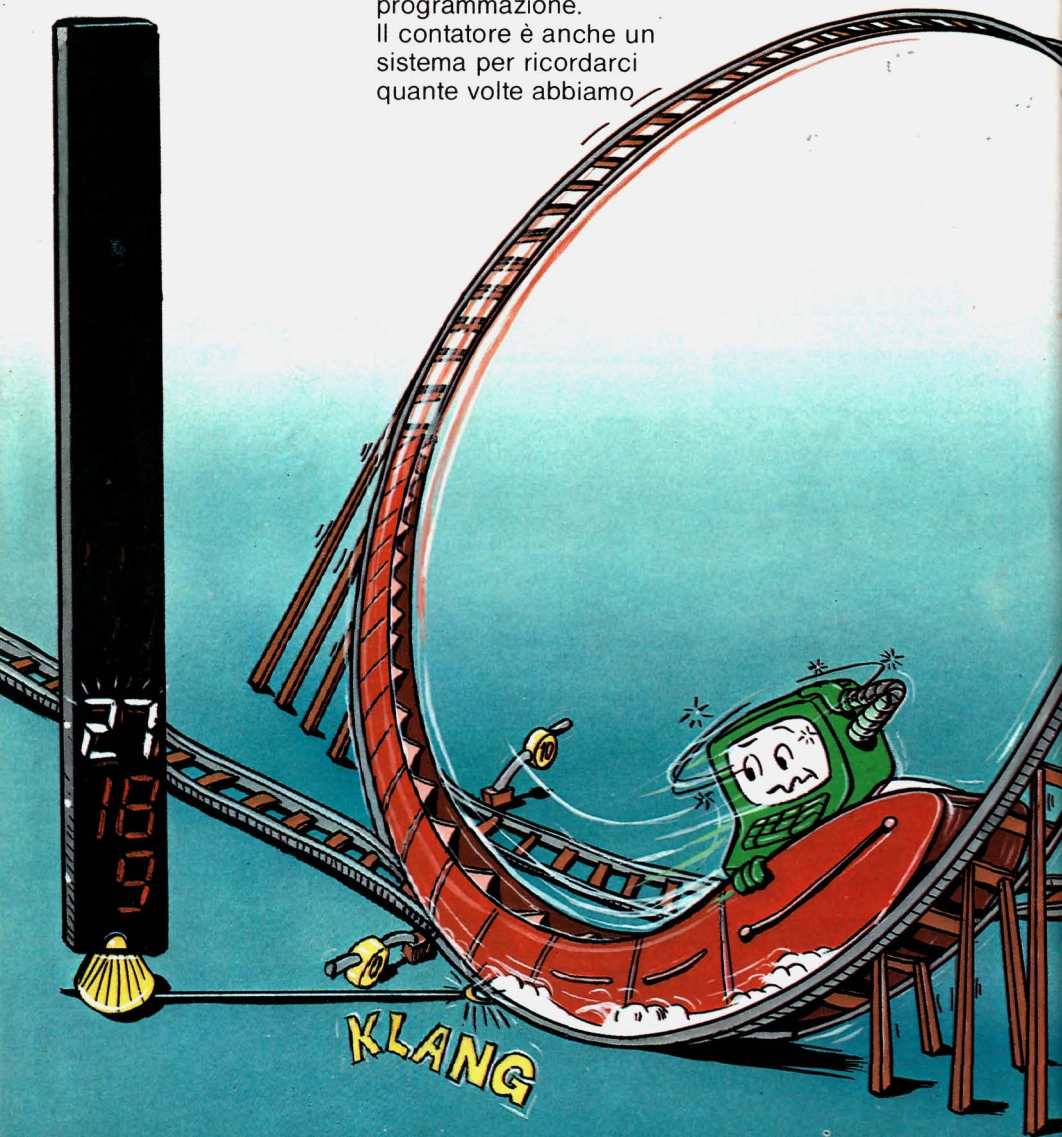
al contatore viene assegnato il valore di partenza della seconda tabellina

stavolta il contatore viene decrementato

a questo punto la tabella del 5 in ordine decrescente sarà stata stampata

Le corrispondenti istruzioni BASIC sono:

```
10 LET NUM = 4
20 PRINT NUM
30 LET NUM = NUM + 4
40 IF NUM <= 40 THEN GOTO 20
50 LET NUM = 50
60 PRINT NUM
70 LET NUM = NUM —5
80 IF NUM >= 5 THEN GOTO 60
90 END
```

Prova a batterle e a eseguirle: se qualcosa non ti è molto chiaro, torna indietro di qualche pagina e rileggi ciò che non hai capito. Cerca, inoltre, di modificare il programma, in modo che stampi altre tabelline pitagoriche, per esempio quelle del 3 e del 7: questo è infatti un ottimo sistema per fare esercizio e pratica di programmazione.
Il contatore è anche un sistema per ricordarci quante volte abbiamo compiuto un'operazione. Scriviamo, ad esempio, un programma che faccia la media di un numero indeterminato di valori, ma per fare la media occorre

conoscere esattamente il numero di valori: come fare? Basta un semplice contatore che si incrementi ogni volta che introduciamo un nuovo numero:

```
10 PRINT " ♥ " : PRINT "MEDIA DEI NUMERI" : PRINT
20 LET SOMMA = 0 : LET NUM = 0
30 PRINT "INDICA FINE CON 0"
40 INPUT "NUMERO ="; N
50 IF N = 0 THEN IF NUM = 0 THEN GOTO 120
60 IF N = 0 THEN GOTO 80
70 LET SOMMA = SOMMA + N : LET NUM =
   NUM + 1 : GOTO 50
80 LET MEDIA = SOMMA/NUM
90 PRINT "LA MEDIA DEI"; NUM; "NUMERI È"; MEDIA
100 INPUT "UN'ALTRA MEDIA? S/N"; R$
110 IF R$ = "S" THEN GOTO 20
120 END
```

FLOW CHART MEDIA DEI NUMERI

INIZIO

SOMMA = 0

CONTATORE = 0

INTRODUCI N

N = 0 — SI / NO

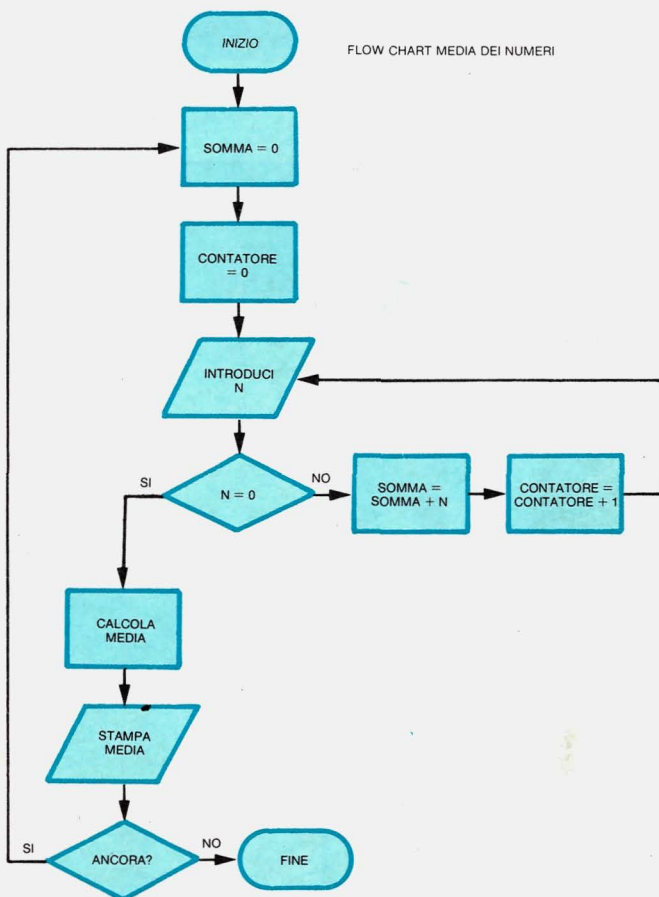
SOMMA = SOMMA + N

CONTATORE = CONTATORE + 1

CALCOLA MEDIA

STAMPA MEDIA

ANCORA? — SI / NO

FINE

VIDEOESERCIZI

Annota nello spazio apposito il risultato da te previsto per ciascun esercizio proposto e poi verificalo con la soluzione del tuo VIC20. Se avrai commesso anche un solo errore ripassa la lezione.

```
10 LET A = - 416.4
20 LET B = INT (A)
30 PRINT B
40 IF A > B THEN PRINT "A > B"
50 IF A < B THEN PRINT "B > A"
```

```
10 LET A = 0 : LET B = . 1 : LET C = A - B
20 PRINT SGN (A)
30 PRINT SGN (B)
40 PRINT SGN (C)
```

```
10 S$ = "*" : C$ = " "
20 C$ = C$ + S$
30 PRINT C$
40 IF C$ = "* * * * * *" THEN END
50 GOTO 20
```

```
10 LET = 100 : LET C = 0 : LET P = 20
20 PRINT C
30 LET C = C + P
40 IF C <= T THEN GOTO 20
50 END
```